Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

Реалізація алгоритму передачі стилю на основі нейронних мереж

Текстова частина до курсової роботи
за спеціальністю «Інженерія програмного забезпечення» 121

Керівник курсової роботи
ст. викл. Бучко О.А.
_____
(підпис)
“___” _____ 2021 р.

Виконав студент ІПЗ МП1
Нгуєн С.Б.В.
“___” _____ 2021р.

Київ 2021

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ
Керівник курсової роботи
_____ст. викл. Бучко О.А.
(підпис)
«____»_____2020 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на дипломну роботу

студенту 5-го курсу, факультету інформатики
Нгуєну Сану Биню Вановичу
ТЕМА: Реалізація алгоритму передачі стилю на основі нейронних мереж
Вихідні дані:
- відео наживо (послідовність зображень відео в реальному часі)

Зміст ТЧ до магістерської роботи:

Зміст
Анотація
Вступ
1. Аналіз предметної області
2. Теоретичні аспекти алгоритму передачі стилю
3. Практичні аспекти алгоритму передачі стилю
4. Реалізація програмного застосунку
Висновки
Список літератури
Додатки

Дата видачі «___» _____ 2020 р. Керівник _____
(підпис)
Завдання отримав _____
(підпис)

Календарний план виконання роботи

Тема: «Реалізація алгоритму передачі стилю на основі нейронних мереж»

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу.	23.10.2020	
2.	Пошук та огляд літератури за темою роботи.	20.12.2020	
3.	Реалізація програми з мінімальним функціоналом.	20.01.2021	
4.	Пошук альтернативних алгоритмів та способи покращення існуючого.	25.02.2021	
5.	Покращення роботи алгоритму.	10.03.2021	
6.	Пошук інших способів реалізації застосунку, дослідження нових алгоритмів.	10.04.2021	
7.	Написання пояснювальної роботи.	02.05.2021	
8.	Створення слайдів для доповіді та написання доповіді.	09.05.2021	
9.	Захист роботи		

Студент *Нгуєн Сан Бинь Ванович*

Керівник *Бучко Олена Андріївна*

“___”_____

ЗМІСТ

Перелік прийнятих скорочень

ПНС – передача нейронного стилю

ЗНМ – згорткова нейронна мережа

НФВ – нефотореалістична візуалізація

ШІ – штучний інтелект

ВР – віртуальна реальність

IoT – internet of things

Анотація

Мета курсової роботи написати застосунок, на вхід якого подається зображення з деяким стилем (наприклад витвір образотворчого мистецтва) та відео в реальному часі, а на виході отримати стилізоване відео. Застосунок працює на iPhone смартфонах. Розглянуто методи покращення нейронного алгоритму, наведені переваги і недоліки різних реалізацій моделі. Створення моделі відбувалося за допомогою PyTorch та Create ML. Для створення застосунку використовувалися системні бібліотек від Apple.

Вступ

Перенесення стилю відносно нова тема, однак швидко стала популярною і знайшла використання майже в усіх сферах діяльності. Використання цього алгоритму не завжди несе суто естетичне задоволення, а ще й приносить користь, наприклад використання перенесення стилю у медицині та космонавтиці. Потенціал нейронного підходу все ще росте, тому у майбутньому будуть винайдені ще варіанти застосування алгоритму. Також збільшення обчислювальних потужностей дозволяє використовувати алгоритм навіть на мобільних приладах, що дозволяє кожному користувачу мати можливість скористатися ним наживо.

Мета цієї роботи простежити розвиток алгоритму передачі стилю, та переглянути нові методи використання зазначеного алгоритму. Також, за ціль покладено створення мобільного застосунку для перевірки працездатності та ефективності роботи алгоритму під час роботи наживо – накладання стилю на відео в режимі реального часу.

У роботі використовується нейронний підхід до створення мережі передачі стилю, а також наведено способи оптимізації та прискорення алгоритму задля роботи на мобільних пристроях. Також досліджені альтернативні методи перенесення стилю та наведені їх недоліки.

Робота складається з чотирьох розділів.

Перший розділ переглядає поточний стан розвитку алгоритму та сфери його використання, такі як медицина, космонавтика, віртуальна реальність, ігрова індустрія, кодування інформації тощо. Наведено короткий опис самого алгоритму та способи його використання.

В другому розділі описується теоретичні відомості про алгоритм, як саме працює алгоритм, з яких компонент він складається, та як оцінюється результат перенесення стилю. Наведені різні типи мереж по відношенню до кількості стилів, яку вони спроможні передати, та альтернативні алгоритми передачі стилю.

Третій розділ присвячено практичним аспектам, а саме тренуванню алгоритму та опис необхідних етапів. В ці етапи входять колекція потрібних даних,

метод оцінки втрат стилю і вмісту при передачі стилю, та описана архітектура мережі.

І в останньому розділі описується створення програмного застосунку під смартфони iPhone та використання сучасних бібліотек для реалізації алгоритму. Наведені та протестовані різні варіанти реалізації застосунку та використання моделей передачі стилю.

Постановка задачі:

а) провести аналіз поточного стану алгоритму передачі стилю; дослідити сфери використання та практичне використання алгоритму;

б) переглянути різні варіанти реалізації алгоритму, навести переваги та недоліки варіацій;

в) побудувати архітектуру алгоритму нейронної передачі стилю, навести етапи для тренування мережі;

г) реалізувати мобільний застосунок, який використовує нейрону мережу для передачі стилю на відео в реальному часі.

Наукова новизна даної роботи полягає у використані досліджених методів реалізації, покращення і оптимізації алгоритму, щоб створити власну архітектуру мережі передачі стилю, щоб створена мережа швидко працювала на мобільному пристрої та накладала стиль в реальному часі з високою роздільною здатністю.

Розділ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Загальні відомості

Передача стилю - це алгоритм комп'ютерного зору, який робить перетворення над одним зображенням так, щоб воно було схоже за стилем на деяке інше еталоне зображення. При такому перетворені вміст зображення не змінюється, однак набуває відмітний стиль другого. Частіше за все за еталоне зображення беруть витвір образотворчого мистецтва, тому отримане вихідне зображення виглядає "намальованим" у стилі заданого стилю.

Рисунок 1.1 – Приклад передачі різних стилів на зображення.

Передача стилю вивчалася протягом декількох десятиліть. Стилізацію зображень та алгоритми винайдені для її здійснення відносять до "нефотореалістичної візуалізації" (англ. "non-photorealistic rendering (NPS)"). Вважається, що термін НФВ був винайдений комітетом SIGGRAPH 1990, який провів сесію під назвою "Нефотореалістичний рендеринг".

Більшість алгоритмів НФВ не є універсальними і розв'язують конкретні проблеми, наприклад, зробити зображення схожим на масляну картину, акварель, ескіз, графіті тощо. Ці алгоритми є прямолінійними і відносно простими в реалізації. Однак питання гнучкості та універсальності продовжувало турбувати

вчених і вже у 2015 році Леон А. Гатіс, Александр С. Екер і Матіас Бездж запропонували новий підхід до задачі передачі стилю [1]. Підхід спирався на глибинному навчанні (англ. "deep learning"), а саме на згорткових нейронних мережах (англ. "convolutional neural network"). Цей підхід передбачав універсальність і гнучкість оскільки нейрона мережа легко навчалася на будь-якому еталонному зображенні. Цей підхід назвали "передача нейронного стилю" (англ. "neural style transfer (NST)"). Нейронна мережа використовується для вилучення статистичних особливостей зображень, такі як вміст (форма, текстура тощо) та стиль, щоб зробити перетворення. За декілька прогонів зображення через мережу можна було отримати те саме зображення з еталонним стилем.

Ранні версії ПНС не були позбавлені недоліків. Одним з них була повільність навчання, оскільки метод розглядав завдання як задачу оптимізації, що вимагає сотень або тисяч ітерацій. Щоб подолати цю неефективність, згодом було розроблено спосіб швидкої передачі нейронного стилю.

Швидка передача стилю також використовує нейронні мережі, але навчає автономну модель, щоб перетворити будь-яке зображення за один прохід. Навчені моделі можуть стилізувати будь-яке зображення лише однією ітерацією, а не тисячами.

1.2 Застосування передачі стилю

Передачі стилю знайшло широке використання у різних сферах. І хоча складно придумати якесь інше застосування інакше від мистецтва, однак винахідники почали використовувати алгоритм і у медицині, космонавтиці та звичайно у комерції.

1.2.1 Мистецтво

Найочевидніше застосування алгоритму передачі стилю можна знайти у образотворчому мистецтві. Використовується цей алгоритм у комерційному мистецтві. Протягом останніх років Крістіз ("Christie’s"), відомий аукціон

предметів мистецтва й розкошу в Лондоні, представляла твори мистецтва штучного інтелекту, одну з яких, “Edmond de Belamy, from La Famille de Belamy”, продали на суму понад 430 000 доларів США [2].

Рисунок 1.2 − “Edmond de Belamy, from La Famille de Belamy” продана за 432 500$ на аукціоні Крістіз у Лондоні [2]

В останні роки завдяки вдосконаленню апаратного забезпечення, яке прискорює ШІ, передача стилю стає можливою і для відео. Застосування алгоритму можна знайти у дизайні, кінематографі та інших видах мистецтва.

1.2.2 Фото та відео редактори

Використання такої особливості як передача стилю робить фото та відео редактори дуже потужними інструментами. А враховуючи гнучкість та ефективність сучасних підходів до глибокого навчання, моделі передачі стилів можуть бути легко вбудовані на мобільні телефони, що дозволяє мобільним застосункам обробляти зображення та відео в режимі реального часу. Це означає, що інструменти для редагування фото та відео професійної якості можуть стати більш доступними та простішими у користуванні, ніж будь-коли раніше. Прикладами таких застосунків є Swoosh, Prisma, Video Star, Painter’s Lens, Looq.

1.2.3 Ігрова індустрія

Під час Конференції розробників ігор 2019 року Google представив Stadia - послугу потокового передавання відеоігор на базі хмар. Однією з основних функцій, включених до цієї демонстрації, була функція передачі стилю в грі, яка автоматично розфарбовувала віртуальний світ за допомогою текстур та різних художніх стилів [3]. Це дало можливість розробникам не турбуватися про візуальне оформлення моделей та поверхонь у розроблених іграх, дозволяючи обійтися без кропіткої роботи дизайнера. Можливість підібрати будь який стиль для своєї гри спростило життя багатьом розробникам одинакам, які давно мріяли створити власну гру, але не мали потрібних навичок в дизайні.

Рисунок 1.3 – Демонстрація роботи Google Stadia – перенесення стилю під гри наживо [3] (зліва зображення стилю, справа ігрова сцена до і після накладення стилю)

1.2.4 Віртуальна реальність

Подібно до ігор, де присутні віртуальні світи, передача стилю може бути застосована у віртуальній реальності. Незважаючи на те, що програми з передачею стилів у ВР все ще знаходяться на стадії досліджень, можливості досить перспективні. Facebook рекламує потенціал передачі стилів, а саме як ВР-технології розповідають та показують історії, ігри, фільми тощо [4].

1.2.5 Аудіо та музика

Передача стилю знайшла своє втілення не тільки серед зображень, але і у аудіо. Використання передачі стилю гарно описана у роботі Ондрея Цифки та співавторів, в якій вони описали як принципи глибинного навчання використовується у передачі тембру і висоти тону між аудіо доріжками [5]. Також відома робота Пратіка Верми і Джуліоса О. Сміта, де вони описали застосування передачі стилю для аудіо спектрограм, яке дозволяє модифікувати аудіо сигнали та генерувати нові звуки [6].

1.2.6 Фізичне моделювання

Використання алгоритму знайшло своє втілення і у фізичному моделюванні. Симулювання рідин, туману, хмар є трудомістким процесом, однак постійно використовуються у кінематографі та розробці ігор. Алгоритм передачі стилю дозволив надавати цим явищам природи привабливий стиль. Докладно про це розповіли Бюнгсу Кім та співавтори у роботі про передачу стилю для диму [7] та у роботі про передачу стилю для рідин [8].

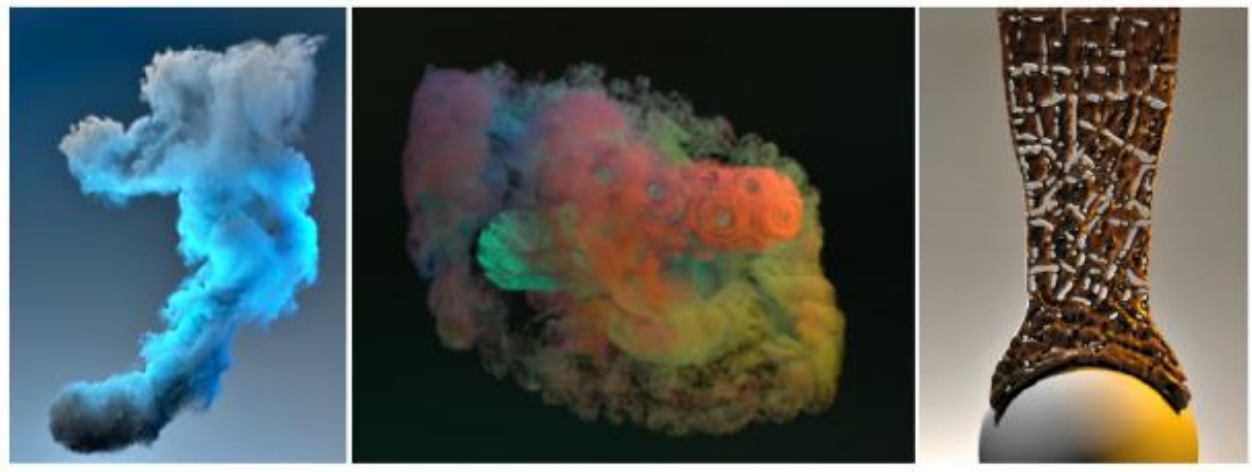


Рисунок 1.4 – Приклад стилізації диму та рідин наведена
в роботі Бюнгсу [8, с.1]

1.2.7 Медицина

У сучасній медицині стало популярним використання методів глибинного навчання під час автоматизованої діагностики захворювань. Нейроні мережі застосовуються і у мамографії. Однак медичні зображення здійснені на різних апаратах сильно відрізняються, оскільки різні постачальники використовують різні домени даних, що потенційно може погіршити результати моделі глибинного навчання. Також це впливає на процес навчання моделі, оскільки зображення не є уніфікованими, тому загальний результат погіршується. Після появи алгоритму передачі стилю, було створено процес для уніфікації стилю зображень створених під час мамографії, що дозволило розробити уніфікований набір даних для тренування моделі та подальше перетворення вхідних зображень до моделі [9]. Таким чином стало можливим виявлення уражень на мамографіях.

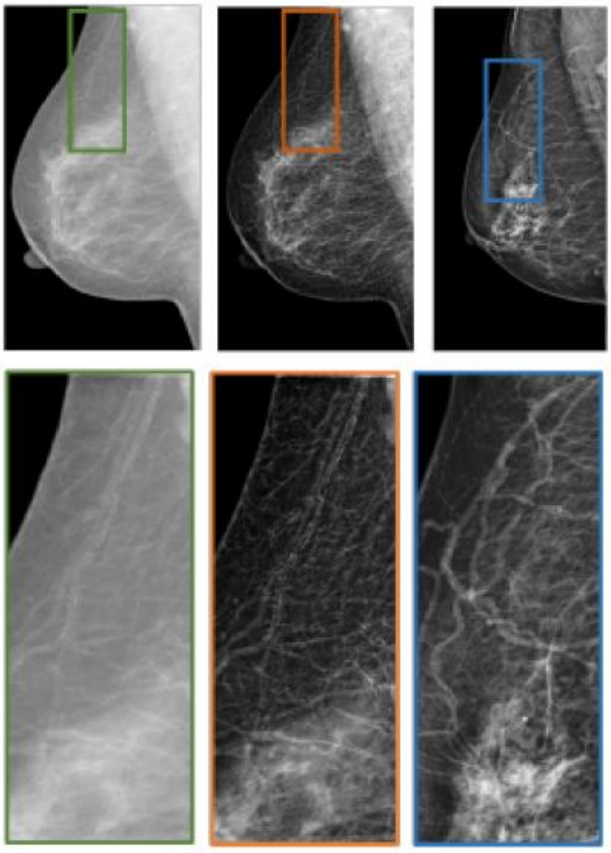


Рисунок 1.5 – Демонстрація роботи ПНС для мамографії (зліва-направо: вхідне зображення, вихідне зображення, зображення стилю (еталонне)) [9, с.6]

1.2.8 Кодування даних

Несподіване застосування алгоритму передачі стилю можна знайти у кодуванні даних, а саме у QR-кодах. Код швидкого реагування ("quick response") є одним із найбільш широко використовуваних двовимірних кодів у всьому світі. Традиційні QR-коди виглядають як випадково розміщенні чорно-білі квадрати, у яких відсутня візуальна семантика та естетичні елементи. У 2020 році було розроблено метод під назвою ArtCoder для створення стилізованих QR-кодів [10], які можуть налаштовуватися під конкретний стиль та є привабливими. Експериментальні результати показали, що стилізовані QR-коди мають високу надійність сканування.

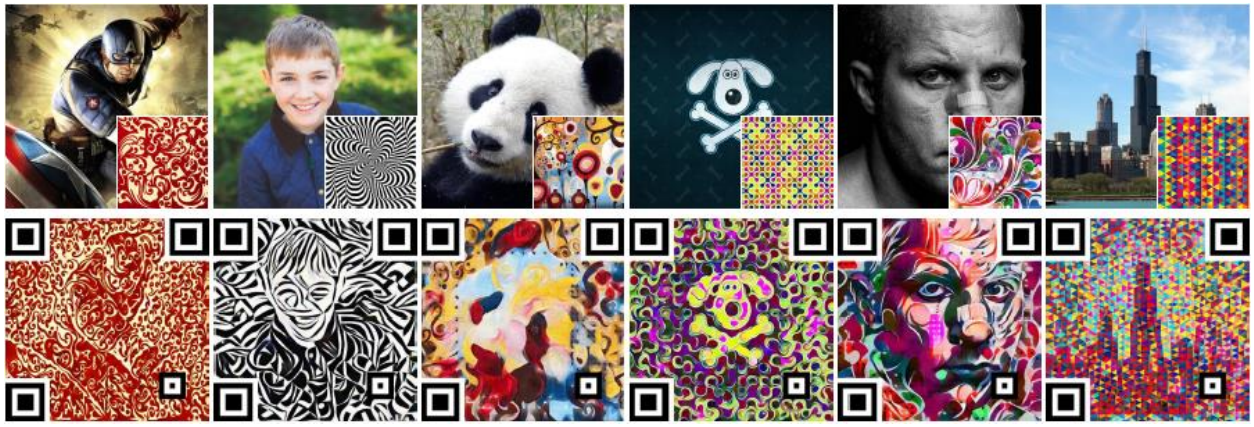


Рисунок 1.6 – Приклади ПНС для QR кодів [10, с.1]

Такі коди вже почали використовуватися у комерційній діяльності. Прикладом такого застосування є відома мережа Макдональдс, яка використовує стилізовані QR-коди у своїх ваучерах та рекламі.

1.2.9 Космонавтика

Цікаве застосування було винайдено для космонавтики: передача стилю використовується для створення наборів даних, зображень з поверхні Місяця, Марса та інших небесних тіл, які використовують для того, щоб тренувати космічні ровери на Землі перед тим, як їх використовувати в реальних місіях [11].

Як еталоні зображення використовують зображенні зібрані під час місії Чан'є-3 і Чан'є-4.

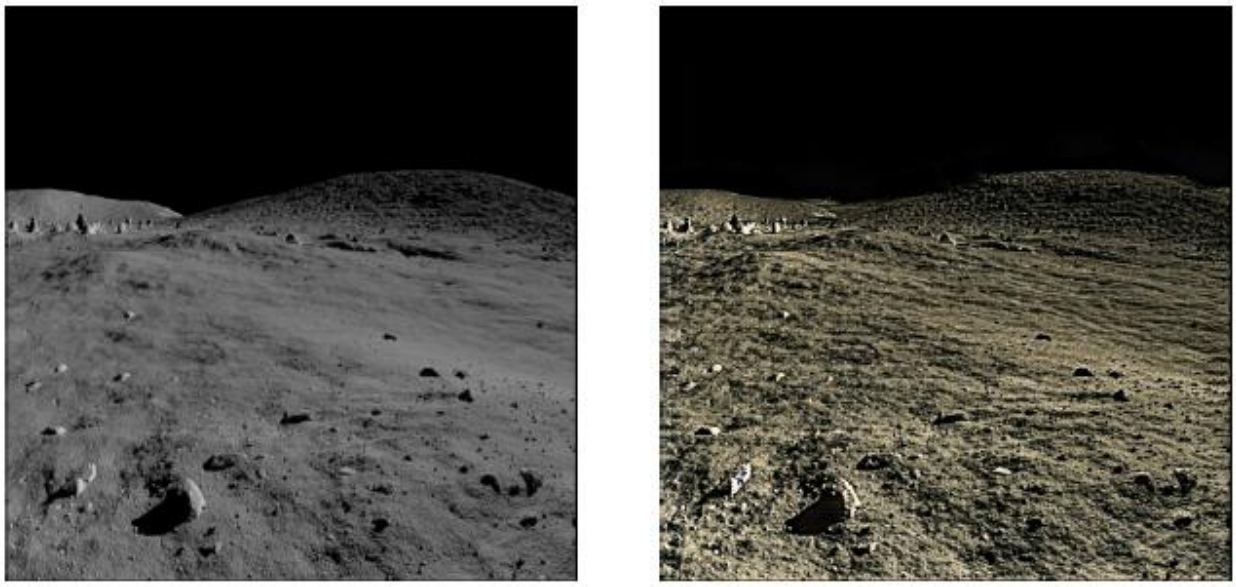


Рисунок 1.7 – Перенесення стилю на симульоване зображення поверхні Місяця (зліва згенероване зображення поверхні Місяця, справа – зображення з накладеним стилем) [11, с. 1]

Розділ 2. ТЕОРЕТИЧНІ АСПЕКТИ АЛГОРИТМУ ПЕРЕДАЧІ СТИЛЮ

2.1 Загальні відомості

У цьому розділі ми розглянемо кілька нейронних підходів до передачі стилів, заснованих на навчанні, та оцінимо їх переваги та обмеження. Хоча існує цілий ряд традиційних методів, ми розглянемо підходи, що використовують нейронні мережі, які стали найпоширенішими у напрямку передачі стилю. Загалом, архітектури глибокого навчання, придатні для передачі стилів, базуються на варіаціях згорткових нейронних мереж (ЗНМ). Щоб простежити ключові відмінності між підходами варто простежити еволюцію алгоритму передачі стилю. Це допоможе зрозуміти які є можливості у передачі стилів.

2.2 Базова структура алгоритму передачі стилю

Навчання моделі передачі стилів вимагає двох мереж: попередньо навченого екстрактора особливостей та мережі передачі.

Попередньо навчений екстрактор особливостей використовується, для того щоб розуміти на які особливості накладати яку частину стилю. Це допомагає мережі передачі стилю більш ефективно та розумно накладати стиль, щоб вихідне зображення виглядало цілісно та не втрачало контент. Деякі шари вчаться витягувати вміст зображення (форму або положення предметів), інші навчаються звертати увагу на фактуру (текстура матеріалів або візерунки природи).

Метод передачі стилів використовує заздалегідь навчену нейронну мережу - пропускає два зображення через неї (вхідне зображення та стилізоване зображення або стилізоване зображення та еталонне зображення) та розглядає результати на декількох шарах та порівнює їх подібність. Зображення, які видають подібні результати на одному шарі попередньо навченої моделі, ймовірно, мають подібний вміст, тоді як зображення, які видають подібні результати на декількох шарах, сигналізують про подібний стиль. Таким чином, щоб перевірити чи стилізоване зображення втратило контент (змінилися форма об'єктів та їх розміщення) достатньо порівняти результати прогону цих зображень на одному шарі,

а якщо треба порівняти як гарно наклався стиль на зображення, треба порівняти результати стилізованого та еталонного зображення на всіх шарах. Тому заздалегідь навчена модель допомагає не тільки знаходити особливості зображень, а й оцінювати наскільки гарно було застосовано алгоритм передачі стилю.

Рисунок 2.1 – Схема оцінки стилізованого зображення за рахунок прогону через згорткову нейрону мережу-екстрактор

Попередньо навчена модель дозволяє нам порівняти вміст і стиль двох зображень, але насправді це не допомагає нам створити стилізоване зображення. Це робота другої нейронної мережі, яку називають мережею передачі. Мережа передачі (стилю) - це мережа перетворення зображень, яка приймає одне зображення як вхідне та виводить інше. Мережі передачі, як правило, мають архітектуру кодера-декодера.

На початку навчання одне або кілька еталонних зображень пропускаються через попередньо навчену модель, а результати на різних шарах зберігаються для подальшого порівняння. Потім звичайні зображення (зображення вмісту) пропускають через мережу, і кожне зображення вмісту проходить через попередньо навчену модель, і також зберігаються результати на різних шарах. Потім зображення вмісту проходить через мережу передачі стилю, яка виводить стилізоване зображення. Стилізоване зображення також пропускається через попередньо навчену модель, а вихідні дані на шарах вмісту і на шарах стилю зберігаються для порівняння (див. Рисунок 2.1).

Якість стилізованого зображення визначається спеціальною функцією втрати, яка зважає як на вміст, так і стиль зображення. Збереженні вихідні дані на шарах (як стилю, так і вмісту) використовуються для порівняння з результатами стилізованого зображення використовуючи вищезгадане правило. Після кожного кроку оновлюється лише мережа передачі. Ваги попередньо навченого екстрактора функцій залишаються фіксованими протягом усього часу. Налаштувавши різні ваги функції втрат, ми можемо навчити модель створювати вихідні зображення із слабкою або сильною стилізацією.

2.3 Огляд архітектури моделі

Існує багато конкретних архітектур нейронних мереж. Розглянемо кілька надійних та популярних архітектур.

2.3.1 Одностильова модель

Джонсон та співавтори 2016 року першими описали створення незалежної нейронної мережі для стилізації зображення за один прохід [12]. У своїй роботі вони використовували попередньо навчену модель VGG16 і відносно невелику мережу кодерів-декодерів як мережу передачі. Цей підхід назвали швидкою передачею стиля і часто використовується на мобільних приладах для застосування стилю наживо. У цьому підході для кожного бажаного стилю навчається одна мережа передачі, що може бути недоліком.

2.3.2 Багатостильова модель

Дослідники Google розширили техніку передачі одного стилю у 2017 році, дозволивши єдиній мережі передачі створювати зображення в декількох стилях і навіть комбінувати більше одного стилю [13]. Їх головним внеском було включення шарів «умовної нормалізації» в мережу, щоб створене стилізоване зображення могло бути обумовлене додатковими параметрами моделі (наприклад

надати одному стилю вагу вдвічі сильнішу ніж другому стилю, щоб вихідне зображення було більш схоже на перший стиль).

Ці мережі отримують на вхід не тільки зображення, а й вектор, який повідомляє мережі, наскільки сильно застосовувати кожен стиль до зображення. Так, наприклад, модель можна було навчити на картинах Малевича, Моне та Далі. Коли настає час стилізувати зображення, користувач може ввести $[1, 0, 0]$ для того щоб застосувати лише стиль Малевича, $[0; 1; 0]$ - тільки Моне, або $[1/3; 1/3; 1/3]$ для комбінування всіх трьох. Це хороший підхід, оскільки він позбавляє від необхідності навчати та зберігати кілька моделей для декількох стилів, а також забезпечує творчу свободу, дозволяючи користувачам змішувати та поєднувати декілька стилів.

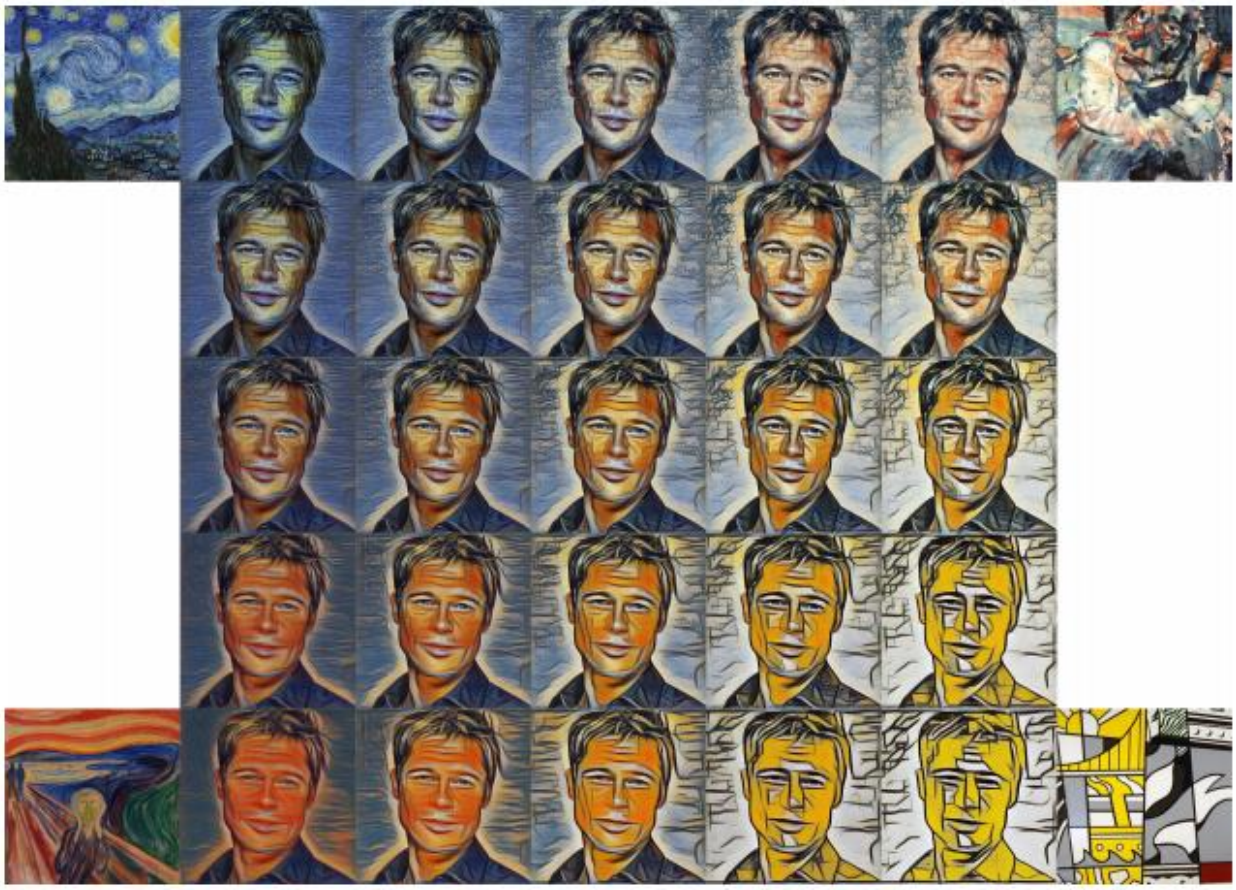


Рисунок 2.2 – Приклад роботи багатостильової моделі ПНС [13, с. 8] (на зображення накладають 4 різних стиля з різними коефіцієнтами)

2.3.3 Модель довільного стилю

Загальною характеристикою як одностильових, так і багатостильових моделей передачі є те, що вони можуть створювати зображення лише в стилях, на яких вони були натреновані. Модель, навчена творам Малевича, не може створювати такі образи, як Далі, не перетренувавши всю мережу. Модель довільної передачі стилю описана Хуангом та співавторами у 2017 відрізняється від цих підходів. Модель довільної передачі стилю бере зображення вмісту та зображення стилю як вхідні дані та виконує передачу стилів за один прохід. Модель вчиться витягувати та застосовувати будь-який стиль до зображення одним махом [14].

2.4 Згорткова нейрона мережа

Згорткові нейронні мережі складаються з шарів перцептронів. Перцептрони це імітація біологічних нейронів. По суті перцептрони є математичними функціями, які отримують та повертають деякі значення. Поведінка кожного перцептрона визначається його вагою. При отримані значень пікселів перцептрони ЗНМ виділяють різні візуальні особливості. Структура ЗНМ запозичена у тварин: схема з'єднання схожа на те, як зорові нейрони зв’язані у корі тварин.

В згорткових нейронних мережах є такі шари: входу, виходу та приховані шари. Приховані шари ЗНМ є різних видів: згорткові (англ. convolutional), агрегувальні (англ. pooling), повноз'єднані (англ. fully connected) та класифікаційний рівень (англ. softmax / logistic layer).

Вхідний рівень у ЗНМ повинен містити дані зображення. Зазвичай дані зображення представлені тривимірною матрицею, наприклад якщо зображення представлене у 3 каналах RGB. Першим кроком буде перетворення зображення в єдиний стовпець. Припустимо, у вас є зображення розміром $28 \times 28 = 784$ пікселів, вам потрібно перетворити його в 784×1 перед подачею на вхід.

Згортковий шар іноді називають шаром екстрактора об’єктів, оскільки особливості зображення витягуються всередині цього шару. Принцип роботи

згорткових шарів полягає в скалярному добутку векторів деякої частини вектор-зображення на вектор-ваги. Коли зображення проходить через ЗНМ, кожен з його шарів генерує карти активації. Карти активації виділяють різні особливості зображення. Кожен з перцептронів приймає ділянку пікселів як вхідний сигнал (вектор-зображення), множить їх на його вагу (вектор-вага), підсумовує їх і запускає через функцію активації. Варто зазначити що згорткові шари також містять активацію ReLU (випрямляч), щоб зробити всі від'ємні значення нульовими.

Перший (або нижній) згортковий шар ЗНМ зазвичай виявляє основні ознаки, такі як горизонтальні, вертикальні та діагональні лінії. Вихід першого шару подається як вхід наступного шару, який витягує більш складні функції, такі як кути та ломані. По мірі проникнення в згорткову нейронну мережу шари починають виявляти функції вищого рівня, такі як об'єкти, обличчя тощо.

Операція множення значень пікселів на ваги та їх підсумовування називається «згорткою» (звідси і назва згорткової нейронної мережі та згорткових шарів). ЗНМ, як правило, складається з декількох згорткових шарів.

Агрегувальний (об'єднуючий) шар використовується для зменшення просторового об'єму вхідного зображення після згортки. Він використовується між двома згортковими шарами. При цьому використовувати звичайну інтерполяцію є не найкращим варіантом, оскільки це буде обчислювально дорого. Замість цього використовують максимальне об'єднання (англ. max pooling). Принцип об'єднання дуже простий: спочатку беруться по декілька суміжних значень (кількість залежить від того, наскільки сильно ми хочемо зменшити об'єм) та залишається лише найбільше. Наприклад якщо ми маємо чотири значення, а залишаємо лише одне то відповідно вихідна кількість значень зменшиться в чотири рази. У шарі об'єднання немає вагів, але він має два гіперпараметри – розмір ядра F – (англ. filter, kernel) та крок S – (англ. stride). Загалом, якщо ми маємо вхідний розмір $W_1 \times H_1 \times D_1$, то

$$W_2 = \frac{(W_1 - F)}{S} + 1,$$
$$H_2 = \frac{(H_1 - F)}{S} + 1, \quad (2.1)$$
$$D_2 = D_1,$$

де W_2, H_2 та W_2 - це ширина, висота та глибина виходу.
Можна побачити що глибина (кількість каналів) залишається не змінною.

Результатом шару об'єднання є узагальнена версія особливостей, виявлених у попередніх шарах. Вони корисні, оскільки невеликі зміни в розташуванні об’єкта на вході призведуть до майже того самого результату як і без зміщення. Ця можливість, спричинена об’єднанням, називається незмінністю моделі до локального зміщення. "У всіх випадках об’єднання допомагає зробити подання приблизно інваріантним щодо невеликих зміщень вхідних даних. Інваріантність до зміщення означає, що якщо ми перемістимо введені дані на невелику величину, значення більшості об'єднаних вихідних даних не зміняться" [15, с.342].

Повноз'єднаний шар включає ваги, які допомагають з'єднувати нейрони одного шару з нейронами іншого шару. Частіше за все він використовується для класифікації зображень, використовуючи особливості високого рівня (останнього згорткового рівня). Повноз'єднаний шар розуміє які "високі" особливості притаманні конкретному класу.

Кінцевий рівень ЗНМ - це класифікаційний рівень. Цей рівень знаходиться відразу після повноз'єднаного шару і приймає його вихідні дані. Цей шар часто має дві різні назви залежно від того чи класифікується багато класів (softmax), чи відбувається двійкова класифікація (logistic). Рівень класифікації видає набір оцінок достовірності (значення від 0 до 1), які вказують, наскільки ймовірно, що зображення належить до "класу". Наприклад, якщо ЗНМ виявляє котів, собак і коней, результат остаточного шару - це можливість того, що вхідне зображення містить будь-яку з цих тварин.

Також виділяють вихідний шар, але основна його задача полягає в тому, що він вказує до якого класу відносяться дані, спираючись на даних, отриманих

на класифікаційному рівні. Вихідний шар може обрати як один найбільш ймовірний клас, так і декілька. Частіше за все на вихідному рівні ми отримуємо той самий вектор що і на класифікаційному рівні, що вже є ймовірностями приналежності до кожного класу.

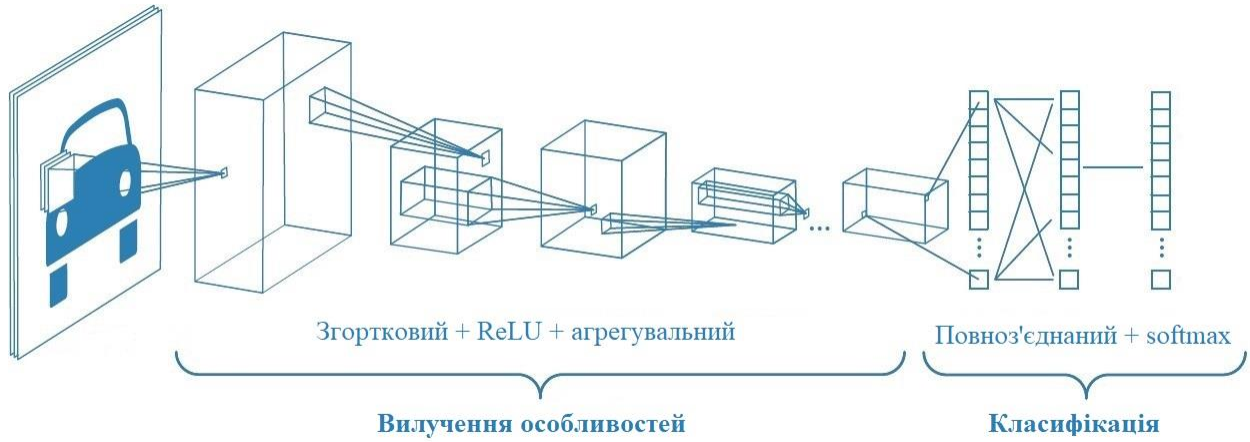


Рисунок 2.3 – Схематичне зображення ЗНМ з різними шарами [16]

В цій роботі будуть використовуватися тільки шари для вилучення особливостей зображення – шари для класифікації будуть видалені.

2.5 Оптимізація та розширення передачі стилю

Нарешті, є кілька оптимізацій та розширень передачі стилів, про які варто згадати. Вони корисні при конкретних випадках використання алгоритму, наприклад в умовах використання на мобільних пристроях, під час відео наживо, або при реалістичного відображення картинки.

2.5.1 Стабільна передача стилю

Перший тип оптимізації це стабілізація передачі стилю. Про це частіше за все згадують, коли передача стилю використовується для відео. Якщо звичайну модель передачі стилю застосувати покадрово для відео, то на вихідному відео можна легко помітити мерехтіння. Навіть невеликі зміни та шум можуть вплинути на результат передачі стилю, що спричинить часову невідповідність для послідовних кадрів. Це може спричиняти зміну кольору та текстури від кадру до

кадру, що порушує консистентність перегляду. Для вирішення цієї проблеми вводять нову вагу, пов’язану із "часовою когерентністю", де два послідовні кадри стилізуються та порівнюються. Це дозволяє тренувати модель створювати ту саму стилізацію для об’єкта, коли він рухається крізь кадр.

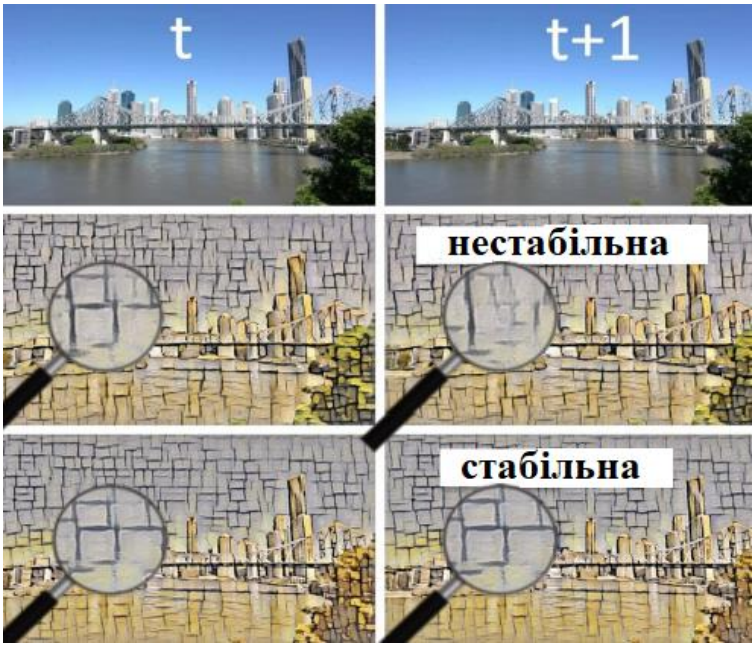


Рисунок 2.4 – Приклад стабільної і нестабільної передачі стилю у момент часу t (ліва колонка) та t+1 (права колонка) [17, с.1]

2.5.2 Кольорова консистентність

Одним з розширень є збереження кольору під час передачі стилю. Інколи варто зберегти кольори об'єктів на вхідному зображенні, але перенести стиль еталонного зображення (мазки, текстуру, візерунки тощо. Це можна зробити різними способами. Одним з них є зміна подання вхідного зображення з RGB на інший колірний простір (наприклад HSV) та застосування передачі стилю лише до каналу яскравості. Також можна застосувати алгоритм передачі кольору до стилізованого зображення.

2.5.3 Фотореалістична передача

Більшість згаданих випадків передачі стилю стосувалося передачі художнього стилю, що є більш простою задачею ніж передача фотореалістичних

характеристик одного зображення на інше. Передачу стилю можна використати для передачі окремих характеристик між двома фотореалістичними зображеннями. Це може бути, наприклад, погода або час доби (застосувати захід сонця до зображення). Передача фотореалістичного стилю, як правило, вимагає модифікації мережі передачі кодера-декодера, щоб зменшити кількість артефактів та підвищенням кількості шарів.

2.6 Швидка передача стилю

Перша версія алгоритму передачі стилю, яка була розроблена Гатісом та співавторами, мала великий недолік: для стилізації кожної картинки потрібно було запускати окремий цикл оптимізації. Система видає не відразу результат, а початковий його варіант, і поступово пропускає початкове зображення декілька разів поки не буде задоволена результатом. Кожне вихідне зображення мінімізує функцію втрат, а тому кожна наступна ітерація намагається зробити зображення ще кращим. Такий процес є громіздким і трудомістким, навіть для графічного процесора. Тому для покращення алгоритму дослідники Джонсон та співавтори винайшли архітектуру, яка стилізувала зображення за один прохід. Тепер такий підхід називають "швидкою передачею стилю" [12]. Цей підхід дав можливість застосовувати стиль в реальному часі та для зображень з високою роздільною здатністю.

2.7 Передача стилю в умовах обмежених потужностей

Частіше за все використання передачі стилю буде відбуватися на приладах користувача, тобто на комп'ютері, мобільному пристрої або на платі IoT. Це означає, що модель мусить ефективно та швидко робити перетворення зображень, особливо коли йде мова про передачу стилю для відео. У таких випадках треба потурбуватися та переконатися, що на пристроях з меншою потужністю все буде працювати безперебійно. Ось декілька порад, щоб переконатись, що модель передачі стилю готова до розгортання.

а) Обріжте мережу, щоб залишити невелику кількість згорткових шарів. Більшість досліджень не обмежують себе в кількості потужностей задля отримання більш привабного результату, тому можуть дозволити собі більшу кількість шарів та параметрів для отримання бажаного результату. Але більшість таких моделей ніколи не були призначені для мобільного використання. Зменшення моделі вплине не тільки на швидкість перетворень, але й на зменшення займаної пам’яті на диску. Для деяких завдань ви можете створити набагато менші мережі, які працюють так само добре, як і великі.

б) Зменшуйте моделі за допомогою квантування, але остерігайтеся падінь точності. Квантування ваг моделі може заощадити купу місця, часто зменшуючи розмір моделі в 4 рази або більше. Однак постраждає точність. Обов’язково ретельно протестуйте квантованні моделі, щоб визначити, чи відповідають вони вашим потребам.

в) Зменшуйте вхідні та вихідні розміри зображень. Хоча може здаватися, що задача потребує модель передачі стилю , яка має можливість приймати фотографії з повною роздільною здатністю, у більшості випадків мобільні пристрої не матимуть достатньої обчислювальної потужності для цього. Загальноприйнято навчати моделі передачі стилів зі скромною роздільною здатністю, а потім застосовувати їх до зображень із більшою роздільною здатністю. Це дозволяє зменшити навантаження на пристрій та пришвидшити процес передачі стилю.

2.8 Альтернативи нейронному підходу

До появи згорткових нейронних мереж для передачі стилю використовувалися інші алгоритми "нефотореалістичної візуалізації". Однак більшість із цих алгоритмів розроблені для конкретних художніх стилів і не можуть бути легко поширені на інші стилі. Поява таких алгоритмів не задовольняла усі потреби, а тому спонукала до створення нових загальних алгоритмів передачі стилю з будь-якого зображення.

Розглянемо алгоритми "художньої візуалізації" (англ. "artistic rendering"), а саме художню стилізацію 2D-зображень, яка називається "художньою обробкою на основі зображень" (англ. "image-based artistic rendering ") [18, с.2].

2.8.1 Візуалізація штрихів

Візуалізація штрихів (англ. stroke-based rendering), подібно художнику, ітеративно накладає штрихи на вхідне зображення. Штрихи підбираються по кольору, та налаштовується їх розмір та частота. Такий алгоритм використовується для таких стилів як олійні картини та ескізів. Однак такий алгоритм підлаштовується лише для одного конкретного стилю і не може імітувати довільний стиль, тому не є гнучким.

2.8.2 Метод регіонів

Метод регіонів (англ. region-based techniques). Візуалізація на основі методу регіонів полягає у сегментації зображення та зафарбовування регіонів у відповідний колір. Регіони можна заповнювати як суцільним кольором так і надавати деяку текстуру. Ступінь сегментації зображення дозволяє контролювати міру деталізації вихідного зображення. Алгоритм "метод регіонів" не здатний імітувати довільний стиль, що робить його не гнучким. Частіше за все його використовують для створення мультиплікаційного ефекту.

2.8.3 Зразкова візуалізація

Зразкова візуалізація (англ. example-based rendering). Метою візуалізації на основі зразків є створення відображення між парою зразкових зображень. Одним із підвидів цього методу Герцман назвав методом аналогій [18, с.2]. Спочатку алгоритм навчають - передають пари зображень і кожному місцю на зображені (це може бути деяка область пікселів, наприклад кернель розміром 3×3) визначають деяку аналогію, яка буде потім замінюватись на вхідних зображеннях. Загалом, аналогії із зображеннями ефективні для різних художніх стилів. Однак

навчальні дані, як правило, дуже важко створити на практиці. Іншим обмеженням є те, що аналогії можуть не враховувати контекст і тому зображення може перетворитися на незрозумілий хаос із шматочків. Іншим підвидом цього методу є використання нейронної мережі.

2.8.4 Обробка та фільтри

Обробка зображень та накладання фільтрів (англ. image processing and filtering)**.** Для створення імітації деякого стилю можна використовувати обробку зображень та накладання фільтрів. Однак для кожного стилю треба створювати свою послідовність елементарних перетворень та окремі фільтри, тому цей спосіб є повільним в реалізації. Не завжди можна створити фільтр для стилю, тому такий спосіб не є універсальним. Однак, цей спосіб залишається найпопулярнішим способом і використовується широко в комерційній діяльності.

Розділ 3. ПРАКТИЧНІ АСПЕКТИ АЛГОРИТМУ ПЕРЕДАЧІ СТИЛЮ

3.1 Етапи навчання моделі

Процес навчання моделі передачі стилю розіб'ємо на три етапи:

а) дані для тренування;

б) архітектура нейронної мережі;

в) функція втрат;

3.2 Колекція даних

Під час класичного сценарію тренування моделі машинного навчання ми зазвичай маємо пари даних $\{X, y\}$, де деяка множина має деякий клас/значення. Це значно спрощує задачу - модель легко може визначити наскільки гарно вона робить передбачення. Модель отримує вхідні дані X та генерує на вихід $\hat{y}$, яке потім порівнюється з вхідним y. Зменшує цю розбіжність, модель покращюється.

Але під час тренування мережі для передачі стилю важко сказати що є y, бо вихідне стилізоване зображення може мати багато варіантів, серед яких чимало буде задовільними. Головними критеріями оцінки мережі звичайно є збереження вмісту і схожість стилю, які можна порівняти за допомогою навченої мережі екстрактора особливостей.

Система передачі стилю складається з таких компонентів:

а) зображення вмісту - використовуються для того, щоб оцінити наскільки гарно мережа зберігає вміст при застосуванні стилю; на сукупність зображень накладається стиль і порівнюється наскільки чітко можна відстежити вміст на стилізованих зображеннях у порівняні з вхідними (у цій роботі використовуватиметься відкрита база зображень COCO, яка містить 5000 зображень і займає приблизно 1 ГБ);

б) зображення стилю (еталоне) - саме стиль цього зображення буде передавати мережа; це може бути я витвір мистецтва деякого відомого автора, або

будь-яке інше фотореалістичне зображення з унікальним стилем, наприклад рентгенівське зображення, або текстура дерева, тощо;

в) вхідне зображення (яке, буде стилізуватися) - це може бути будь-яке зображення або просто шум.

3.3 Архітектура

Наступний крок це побудова архітектури мережі. На вхід буде подаватися зображення, яке по суті є набором значень $3 \times N \times N$ (3 канали RGB та розмір зображення N). Вхід і вихід мережі - це зображення, тому очевидно що вхідний розмір даних і вихідний будуть співпадати. При класичному підході на виході частіше за все ми маємо деяку класифікацію (наприклад, бінарну - чи відноситься зображення до заданого клас чи ні, або класифікаційну - до якого саме класу відноситься зображення), тому розмір даних на виході зменшується. У випадку передачі стилю це не так: розмір поступово зменшується, а потім збільшується. Тому відразу можна згадати такий клас нейронних мереж як U-подібні.

У таких мережах використовуються згорткові шари які спочатку зменшують (згортають) дані, а потім збільшують (розгортають). Тому на виході дані будуть мати вхідний розмір. Під час згортання дані зменшується у ширину та висоту, але збільшуються у глибину - це спричиняє зменшення просторової інформації та розширює набір особливостей. Під час розгортання - збільшується роздільна здатність даних і зменшується кількість каналів (глибина).

Першою і найвідомішою архітектурою серед цього сімейства моделей є U-net. Вона буда описана у роботі Роненберга "U-Net: згорткові мережі для біомедичної сегментації зображень" у 2015 році [19]. Особливою ознакою цієї мережі є нелінійний прохід даних. Вводяться додаткові шари конкатенації, які об’єднують дані з однакових рівнів даних. Цей підхід був попередником для моделі ResNet, і був використаний і для передачі стилів.

У архітектурі мережі передачі стилю будемо використовувати такі види шарів (назви будуть продубльовано з оригінальними назвами в коді):

а) згорткові шари (ConvLayer) та розгорткові шари (UpsampleConv-Layer);

б) шари нормалізації екземпляру (InstanceNorm2d) - було введено Уляновим та співавторами на заміну "batch normalization" (нормалізація застосовується до кожного каналу даних, наприклад, RGB, що значно покращує результати застосування стилю та якість мережі);

в) випрямляч (англ. ReLU - rectified linear unit);

г) залишкові шари (ResidualBlock).

До нормалізації екземплярів використовувався стандартна норма (BN - batch normalization). Стандартна норма обчислює одне середнє і один розподіл на сукупність даних і нормалізує всі зображення в партії за цими значеннями. Нормалізація екземплярів обчислює одне середнє і один розподіл для кожного каналу для кожного зображення, що значно покращує загальні результати [20].

Суцільну архітектуру, розташування шарів та їх параметри, можна переглянути на Додатку А. Загальний шлях виглядає наступним чином: спочатку зображення проходить низхідний шлях через 3 рівні згорткових шарів (згортковий → нормалізація екземпляру → випрямляч), потім через три рівня залишкових шарів (згортковий → нормалізація екземпляру → випрямляч → конкатенація), а потім висхідний шлях через 3 рівня розгорткових шарів (згортковий → нормалізація екземпляру → випрямляч). На вхід подіється зображення розміром $512 \times 512 \times 3$. Задана архітектура гарно працює під час живої передачі стилю (застосування стилю в реальному часі).

3.4 Функція втрат

Щоб розуміти наскільки гарно натренована мережа нам знадобиться метод її оцінки - функція втрат. Далі формалізуємо процес підрахунку втрат. Цей процес розбивається на дві частини - підрахунок втрат вмісту та втрат стилю.

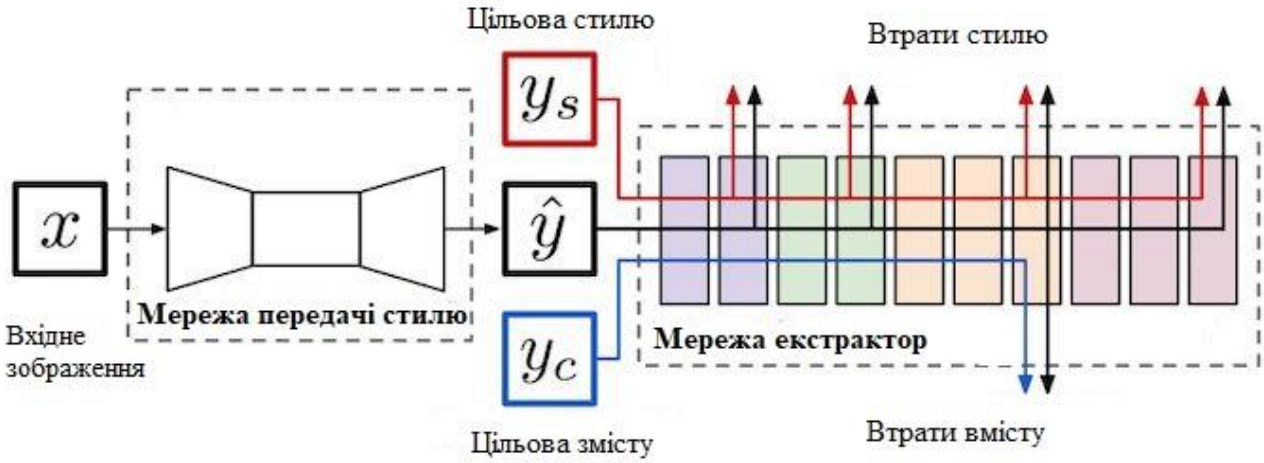


Рисунок 3.1 – Схематичне зображення процесу оцінки втрат стилю і вмісту

Алгоритм оцінки втрат стилю:

1) пропустити оригінальне зображення через заздалегідь навчену мережу (наприклад, VGG19);

2) для кожного згорткового шару обрахувати матрицю Грама;

3) повторити кроки 1 та 2 для стилізованого зображення (вихід мережі передачі стилю);

4) обрахувати середнє квадратичне відхилення між парами матриць Грама на кроках 2 і 3;

5) просумувати відхилення на 4 кроці. Отримаємо втрати стилю.

Алгоритм оцінки втрат вмісту:

1) пропустити оригінальне зображення через заздалегідь навчену мережу (наприклад, VGG19);

2) вилучити значення, наприклад з 3-го згорткового шару;

3) повторити кроки 1 та 2 для стилізованого зображення (вихід мережі передачі стилю);

4) обрахувати середнє квадратичне відхилення між кроками 2 і 3;

5) отримаємо втрати вмісту.

Сума втрати вмісту і втрати стилю є загальною оцінкою втрат.

Варто зазначити що матриця Грама це квадратна симетрична матриця яка утворюється під час добутку матриці та транспонованої матриці. У випадку згорткових шарів - саме матриця вагів буде використана у добутку. Припустимо ми маємо згортковий шар розміром $512 \times 30 \times 30$. Щоб використати цей шар треба її сплюснути до двовимірного представлення - перетворимо її на матрицю 512×900. Далі ми знаходимо транспоновану матрицю і здійснюємо множення - 512×900 матриця та 900×512 матриця. В результаті отримуємо матрицю 512×512. І таку матрицю легко використати під час підрахунку втрат.

Використання матриці Грама як спосіб оцінки втрат має наступні переваги. Матриця Грама допомагає знехтувати просторовим розміщенням елементів у еталонного зображення. Перенесення елементів з еталонного зображення не є цілю, тому ми ігноруємо вміст зображення і концентруємо увагу на колір і текстуру зображення. Якщо як еталоне зображення використовується масляна картина, то модель зосереджується на мазках художника. Стискання просторових даних та підрахунок матриці Грама гарно з цим справляються.

Матриця Грама дозволяє отримати представлення які особливості зображення є окремим елементом або сукупністю інших особливостей. Оскільки використовується заздалегідь навчена модель, яку у свою чергу вирізняє ці особливості, то розуміння як накладати стиль на кожен елемент є ціною інформацію. Інакше стиль накладався б хаотично і не мав консистентності.

Варто зазначити ще один важливий аспект підрахунку втрат стилю та вмісту - це важливість кожного фактору. Під час тренування можна задавати коефіцієнти для цих двох метрик - наприклад надавати стилю більший коефіцієнт, що зробить зображення сильніше стилізованим, або надавати вагу вмісту - що залишить вміст більш виразним. Ці коефіцієнти безпосередньо домножуються на етапі підрахунку втрат.

Після отримання значень втрат, застосовуються методи оптимізації. В даній роботі використовується класичний та поширений метод Adam ("adaptive moment estimation").

Розділ 4. РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАСТОСУНКУ

4.1 Загальні відомості

Екосистема техніки та програмного забезпечення компанії Apple стрімко росте, і все більше користувачів використовують здобутки компанії в повсякденному житті. Однією з цілей цієї роботи було показати наскільки гарно справляється алгоритм передачі стилю в умовах обмеженої кількості ресурсів - а саме потужностей мобільних процесорів. Хоча смартфони компанії Apple залишаються одними з найсучасніших та найпотужніших, однак це не може зрівнятися з потужними процесорами, особливо графічними, для стаціонарних комп'ютерів. Тим не менш, навіть з такими можливостями, смартфони iPhone легко справляться з поставленою задачею. Для розробки під операційну систему iOS (операційна система смартфонів iPhone та деяких планшетів iPad) використовується нативна мова програмування Swift. Ця мова є відносно новою і була розроблена Apple для розробки програмного забезпечення саме для її пристроїв. Як середовище розробки використовується Xcode, яка дозволяє писати додатки для усіх пристроїв Apple.

Xcode має допоміжні програми, наприклад Create ML, який дозволяє тренувати деякі нейронні мережі на комп'ютерах з операційною системою MacOS. Серед цих моделей можна знайти й мережу передачі стилю. В цій роботі будуть порівнюватися результати різних мереж і від Apple також. Варто зазначити, що програма Create ML допомагає натренувати мережу навіть не інженерам за рахунок зручного користувацького інтерфейсу. Проте, зміни у структурі самої мережі неможливі, тому гнучкість такої мережі обмежена заздалегідь визначеними параметрами інженерами Apple.

Компанія Apple притримується усіх сучасних трендів, та підтримує власну бібліотеку для розгортання нейронних мереж. Для цього вона використовує багато різних внутрішніх бібліотек. Остання назва головної бібліотеки, яка підтримує усі останні зміни у сфері нейронних мереж, має назву CoreML. Ця бібліотека потужна і підтримує різні сценарії розгортання мереж. Головна перевага, що

багато коду вже написано інженерами Apple, і код оптимізований до різних процесорів і пристроїв Apple. Робота мереж можлива як на CPU так і на GPU (графічних процесорах). Саме робота на графічних процесорах робить роботу складних нейронних мереж можливою на мобільних пристроях Apple.

Оскільки нейроні мережі працюють на графічних процесорах, то є проміжний прошарок взаємодії з ними - інтерфейс для взаємодії з апаратним забезпеченням. У Apple було створено власну специфікацію та мову для взаємодіє з графічним ядром - мову Metal. Metal - це низькорівнева мова. Мова Metal прийшла на заміну OpenGL у 2014 році, і Apple почала просувати її у всіх своїх напрямках розробки - розробка ігор, створення нейронних мереж, доповнена та віртуальна реальність тощо. Metal багато всього перейняв у свого попередника, і має багато схожого. Щоб написати програму на Metal треба створити так званий "шейдер", який запускається на графічному процесорі. Сам синтаксис дуже схожий на мову C.

В цій роботі розроблено додаток, який об’єднує декілька різник функцій.

а) використання готових моделей CoreML, які були натреновані за допомогою застосунку Create ML;

б) використання моделей PyTorch, які були натреновані та експортовані у натині моделі CoreML;

в) використання експортованих вагів PyTorch моделей та відтворення архітектури на iOS пристроях за допомогою Metal та супутніх бібліотек.

Написання власної нейронної мережі на мові Metal є досить простою задачею, оскільки багато коду для нейронних мереж вже включено до пакету MetalKit, і розробники можуть швидко відтворити архітектуру вже натренованих моделей. Експорт вагів також є простою задачею, однак треба зважати на різні формати різних бібліотек. В даній роботі використовувалися моделі тільки бібліотеки PyTorch, проте можна використовувати будь які бібліотеки - головне мати можливість отримати ці ваги в деякому форматі.

4.2 Create ML

Створення моделей за допомогою застосунку Create ML є дуже простою задачею. Окрім мережі передачі стилю, цей застосунок передбачає інші нейроні мережі для класифікації зображень, текстів, звуків і активності користувачів, знаходження об'єктів на зображені, система рекомендацій тощо. Графічний інтерфейс застосунку є інтуїтивно зрозумілим і допомагає натренувати моделі навіть недосвідченим користувачам. На прикладі моделі для перенесення стилю, можна побачити наступні інструменти застосунку (див. Рисунок 4.1).

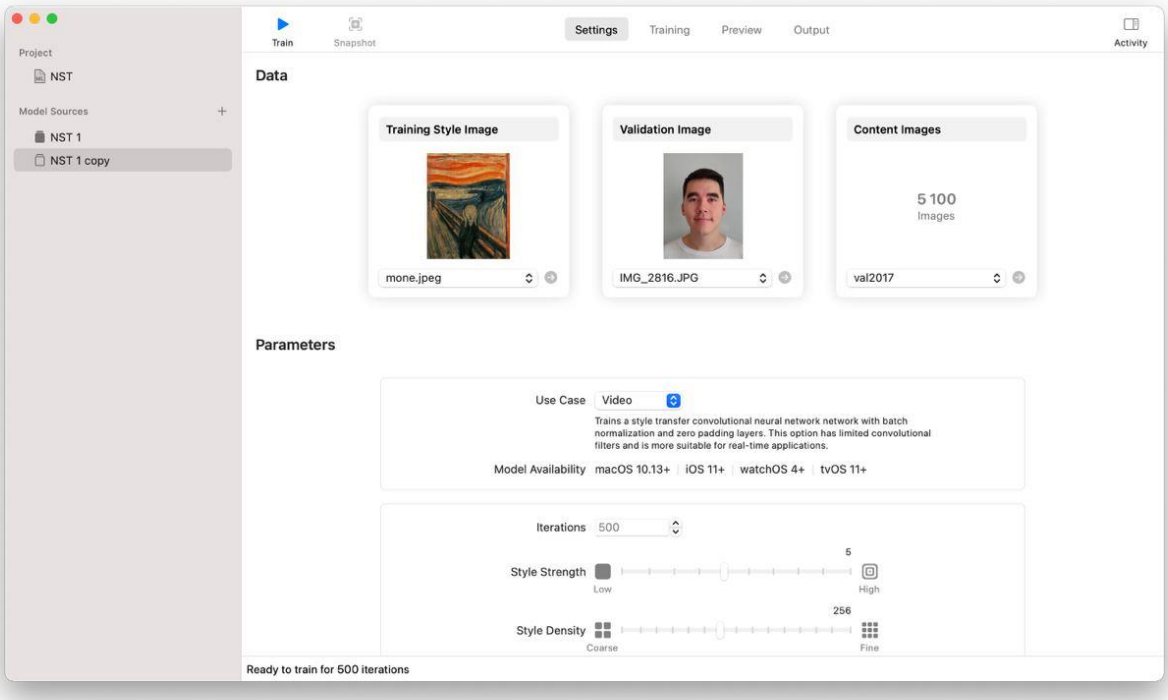


Рисунок 4.1 – Графічний інтерфейс програми Create ML

Перше це вхідні дані - для нейронної мережі для перенесення стилю їх 3 складові: зображення стилю, набір зображень вмісту (наприклад можна використати власний набір зображень для тренування) та власне зображення для валідації результату. Окрім вхідних зображень, застосунок надає можливість обрати силу накладання стилю та щільність (наскільки виразно буде видно окремі елементи стилю). Ще одною перевагою є можливість обрати для чого буде використана модель - для відео чи для фото. У разі відео - система скоротить кількість

шарів, щоб мережа могла працювати під час роботи наживо. Саме такі моделі підходять для роботи на мобільних пристроях в реальному часі. Під час тренування моделі можна побачити графіки зміни втрат стилю і вмісту у реальному часі, і переглядати як стиль накладається на кожному кроці (див. Рисунок 4.2). Також можна зберегти модель посеред тренування, наприклад, якщо модель задовольняє поставленій задачі, або для порівняння з наступними ітераціями. Зберігається модель в форматі .mlmodel, який був створений Apple для представлення нейронних мереж.

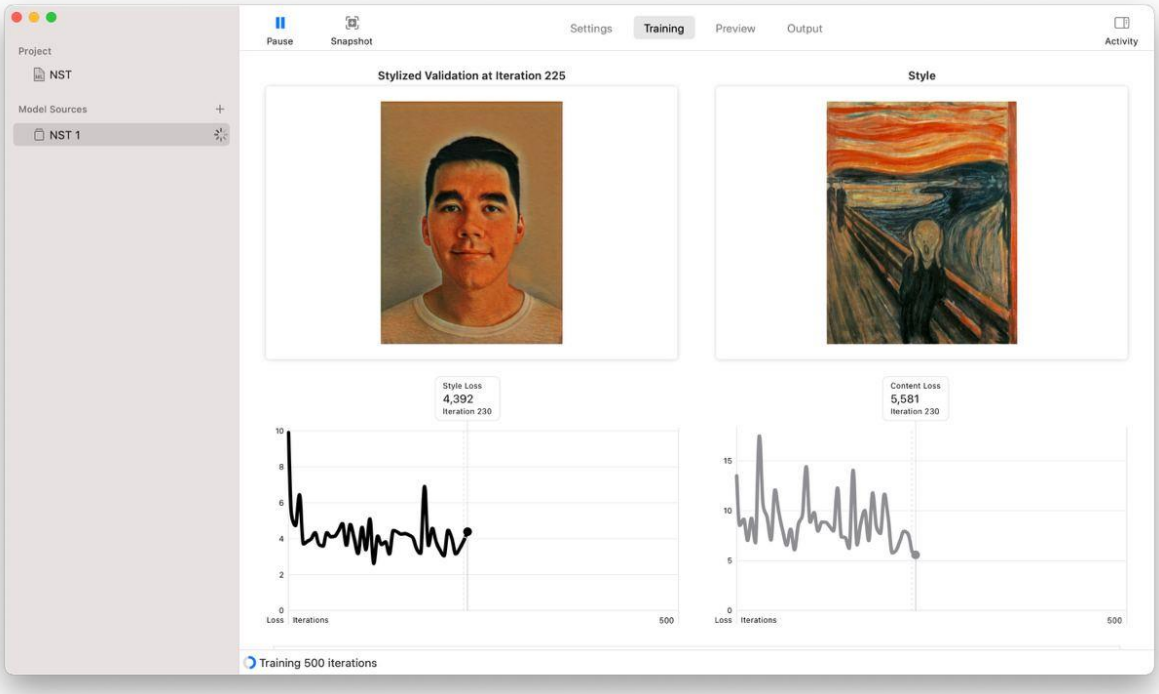


Рисунок 4.2 – Процес тренування мережі ПНС у програмі Create ML

У цій роботі натреновано декілька мереж за допомогою Create ML, щоб продемонструвати роботу з цією системою та порівняти результати з іншими варіантами мереж передачі стилю. З результатами мереж створених на Create ML можна ознайомитися в Додатку Б.

Варто відзначити, оскільки мережа тренується на комп'ютерах Apple, швидкість тренування не дуже швидка. Для того щоб натренувати одну модель передачі стилю може знадобитися від 5 до 20 хв (в залежності від моделі і потужності комп'ютера).

4.3 PyTorch

Бібліотека PyTorch одна з багатьох відомих бібліотек для машинного навчання написані для мови програмування Python. За допомогою цієї бібліотеки можна побудувати будь-яку модель машинного та глибинного навчання. За допомогою PyTorch можна побудувати мережі на основі вже натренованих, що і буде використовуватися в цій роботі, а саме будуть використовуватися вже навчена модель VGG16, та оптимізаційний метод Adam. За допомогою же визначених шарів (згорткових, агрегуючих тощо) буде створена мережа передачі стилю. Щоб використати вже готову модель, є декілька варіантів.

Перший варіант, експортувати PyTorch модель у CoreML модель. До переваг такого варіанту можна віднести простоту впровадження системи у будь-яке програмне забезпечення Apple (застосунки для iOS, macOS тощо) та можливість зашифрувати вміст моделі, щоб захистити її від викрадення. Недоліків у такому варіанті не багато, і найбільший з них - це неможливість експортувати деяких шарів (тих які власноруч створив користувач). Щоб вирішити таку проблему, знадобиться скористатися другим варіантом.

Другий варіант - це експортувати тільки ваги готової моделі. Потім ці ваги треба буде імпортувати в застосунок з вже відтвореною архітектурою мережі. Перевагою такого варіанту є те, що можна створити будь-яку мережу, обмежень не має - користувач може створювати власні шари, впорядковувати їх як завгодно та створювати складну розгалужену систему. У цьому всьому може допомогти Metal, який має деякі вже визначені шари та дозволяє написати власні за потреби. Однак недоліків у такого варіанту теж багато. При такому варіанті треба власноруч відтворювати архітектуру мережі, не має способу захисту вагів від викрадення (хоча використати ці ваги буде складніше, бо крадій не буде знати точної архітектури системи і до якого саме шару відносяться вказані ваги), і треба гарно знатися у роботі Metal.

В цій роботі розглянуті обидва варіанти, однак перевагу отримав перший варіант саме за зручність. Для більш складних архітектур рекомендується

використовувати другий варіант за необхідності, проте в Apple намагаються покрити всі можливі сценарії, щоб нівелювати недоліки використання CoreML. З результатами цих двох варіантів можна ознайомитися в Додатку В.

4.4 Огляд застосунку

Під час даного дослідження було створено iOS застосунок для демонстрації роботи створених моделей та перевірки їх працездатності у реальних умовах (а саме наскільки гарно та ефективно вони працюють в умовах обмеженої кількісті обчислювальних потужностей). Як критерій роботи було виставлено можливість накладати стиль наживо на відео з частотою 30 кадрів на секунду.

Застосунок допомагає знімати відео на камеру смартфона та накладає обраний стиль в реальному часі. У застосунку представлено приблизно 10 різних стилів, які мають по 2-3 варіації (Create ML, PyTorch, різні параметри моделі). Знімати відео можна як на задню, так і на передню камеру. Відео знімається в максимально можливій роздільній здатності, однак обрізається до квадратного формату. Застосунок підтримує три варіанти імпорту моделей передачі стилю: Create ML модель через .mlmodel файл, PyTorch модель через .mlmodel файл, та архів вагів PyTorch моделі у форматі .bin окремо для кожного шару мережі.

Застосунок складається з таких частин: графічний інтерфейс системи, модуль захоплення відео, та модуль накладання стилю на відео. Для написання модулів використовувалися системні бібліотеки iOS, а саме:

а) UIKit - бібліотека для створення графічного інтерфейсу для iOS пристроїв;

б) AVFoundation - бібліотека для роботи з відео та аудіо;

в) CoreMedia - низькорівнева бібліотека для роботи з медіа представленнями;

г) Combine - бібліотека для реактивного програмування;

д) Metal - низькорівнева бібліотека для роботи з графічним процесором та написання нейронних мереж і шейдерів;

ж) CoreML - високорівнева бібліотека для роботи з нейронними мережами.

Цілком вся програма написана на мові програмування Swift версії 5, окрім деяких частин (шейдери написані на мові Metal та деякі структури написані на мові Objective-C). Як середовище розробки використовувався Xcode версії 12.4. Передумовою роботи застосунку є версія iOS 14 і вище.

З графічним інтерфейсом програми можна ознайомитися у Додатку Г.

Висновки

У ході виконаної роботи було створено власну мережу передачі стилю та запропоновано архітектуру для швидкої передачі стилю, яка вправно працює для відео наживо.

Також було створено мобільний застосунок під операційну систему iOS для мобільних пристроїв. Створений застосунок дав можливість використовувати підготовлені моделі передачі стилю для накладання стилю під час зйомки відео в реальному часі. Даний застосунок продемонстрував можливість створення нейронної мережі, яка ефективно та безперебійно працює на мобільних пристроях. Підготовлені моделі передачі стилю було оптимізовано саме для роботи наживо з відео.

Для використання та створення мереж використовувались як вже готові високорівневі рішення для розгортання моделей (PyTorch, Core ML, Create ML), так і низькорівневі рішення – створення архітектури за допомогою низькорівневої мови Metal, яка дозволяє працювати з графічним ядром.

Було досліджено поточний стан алгоритму та виявлено цікаві випадки використання алгоритму – у медицині, космонавтиці, кодуванні даних тощо.

Наступними кроками застосування алгоритму може бути створення практичних застосунків для мобільних пристроїв – наприклад, наведені випадки використання у медицині. Також потенційно можна перенести навчання мережі передачі стилю на мобільний пристрій, щоб користувач мав можливість натренувати мережу одразу на своєму приладі та використати відразу на цьому пристрої. Таку можливість можна здійснити за допомогою вже наявних бібліотек для навчання на мобільних пристроях (наприклад Compute ML від компанії Apple).

Список літератури

1. Gatys L. A. A Neural Algorithm of Artistic Style [Електронний ресурс] / Leon A. Gatys, Alexander S. Ecker, Matthias Bethge. — [Б. м. : б. в.]. — 16 с. — (Препринт ; arXiv:1508.06576v2). — Режим доступу: https://arxiv.org/abs/1508.06576 (дата звернення: 30.04.2021). — Назва з екрана.

2. Cohn G. AI Art at Christie's Sells for $432,500 (Published 2018) [Електронний ресурс] / Gabe Cohn // The New York Times. — Режим доступу: https://www.nytimes.com/2018/10/25/arts/design/ai-art-sold-christies.html (дата звернення: 30.04.2021). — Назва з екрана.

3. Nelva G. Watch Google Stadia in Action in Impressive Demo Videos; Launch Coming in 2019 [Електронний ресурс] / Giuseppe Nelva // Twinfinite. — Режим доступу: https://twinfinite.net/2019/03/google-stadia-action-demo-videos/ (дата звернення: 30.04.2021). — Назва з екрана.

4. Using AI for new visual storytelling techniques in VR - Facebook Engineering [Електронний ресурс] // Facebook Engineering. — Режим доступу: https://engineering.fb.com/2017/07/26/virtual-reality/using-ai-for-new-visual-storytelling-techniques-in-vr/ (дата звернення: 30.04.2021). — Назва з екрана.

5. Self-Supervised Vq-Vae For One-Shot Music Style Transfer [Електронний ресурс] / Ondrej Cifka [та ін.]. — [Б. м. : б. в.], 2021. — 5 с. — (Препринт ; arXiv:2102.05749). — Режим доступу: https://arxiv.org/abs/2102.05749v1 (дата звернення: 30.04.2021). — Назва з екрана.

6. Verma P. Neural Style Transfer for Audio Spectrograms [Електронний ресурс] / Prateek Verma, Julius O. Smith. — [Б. м. : б. в.], 2018. — 3 с. — (Препринт ; arXiv:1801.01589v1). — Режим доступу: https://arxiv.org/abs/1801.01589 (дата звернення: 30.04.2021). — Назва з екрана.

7. Transport-Based Neural Style Transfer for Smoke Simulations [Електронний ресурс] / Byungsoo Kim [та ін.]. — [Б. м. : б. в.], 2019. — 11 с. — (Препринт ; arXiv:1905.07442v2). — Режим доступу: https://arxiv.org/abs/1905.07442 (дата звернення: 30.04.2021). — Назва з екрана.

8. Lagrangian Neural Style Transfer for Fluids [Електронний ресурс] / Byungsoo Kim [та ін.]. — [Б. м. : б. в.], 2020. — 10 с. — (Препринт ; arXiv:2005.00803v1). — Режим доступу: https://arxiv.org/abs/2005.00803v1 (дата звернення: 30.04.2021). — Назва з екрана.

9. mr2NST: Multi-Resolution and Multi-Reference Neural Style Transfer for Mammography [Електронний ресурс] / Sheng Wang [та ін.]. — [Б. м. : б. в.], 2020. — 9 с. — (Препринт ; arXiv:2005.11926v1). — Режим доступу: https://arxiv.org/abs/2005.11926v1 (дата звернення: 30.04.2021). — Назва з екрана.

10. Su H. An End-to-end Method for Producing Scanning-robust Stylized QR Codes [Електронний ресурс] / Hao Su, Jianwei Niu, Xuefeng Liu ; ред. Q. Li. — [Б. м. : б. в.], 2020. — 12 с. — (Препринт ; arXiv:2011.07815v1). — Режим доступу: https://arxiv.org/abs/arXiv:2011.07815v1 (дата звернення: 30.04.2021). — Назва з екрана.

11. Curtó J. Cycle-consistent Generative Adversarial Networks for Neural Style Transfer using data from Chang'E-4 [Електронний ресурс] / J. de Curtó, R. Duvall. — [Б. м. : б. в.], 2020. — (Препринт ; arXiv:2011.11627v1). — Режим доступу: https://arxiv.org/abs/2011.11627v1 (дата звернення: 30.04.2021). — Назва з екрана.

12. Johnson J. Perceptual Losses for Real-Time Style Transfer and Super-Resolution [Електронний ресурс] / Justin Johnson, Alexandre Alahi, Li Fei-Fei. — [Б. м. : б. в.], 2016. — 18 с. — (Препринт ; arXiv:1603.08155v1). — Режим доступу: https://arxiv.org/abs/1603.08155v1 (дата звернення: 30.04.2021). — Назва з екрана.

13. Dumoulin V. A Learned Representation For Artistic Style [Електронний ресурс] / Vincent Dumoulin, Jonathon Shlens, Manjunath Kudlur. — [Б. м. : б. в.], 2017. — 26 с. — (Препринт ; arXiv:1610.07629v5). — Режим доступу: https://arxiv.org/abs/1610.07629 (дата звернення: 30.04.2021). — Назва з екрана.

14. Huang X. Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization [Електронний ресурс] / Xun Huang, Serge Belongie. — [Б. м. : б. в.], 2017. — 11 с. — (Препринт ; arXiv:1703.06868v2). — Режим доступу: https://arxiv.org/abs/1703.06868 (дата звернення: 30.04.2021). — Назва з екрана.

15. Semi-Supervised Learning (Adaptive Computation and Machine Learning). — [Б. м.] : The MIT Press, 2006. — 498 с.

16. Saha S. A Comprehensive Guide to Convolutional Neural Networks—the ELI5 way [Електронний ресурс] / Sumit Saha // Medium. — Режим доступу: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (дата звернення: 30.04.2021). — Назва з екрана.

17. Characterizing and Improving Stability in Neural Style Transfer [Електронний ресурс] / Agrim Gupta [та ін.]. — [Б. м. : б. в.], 2017. — 10 с. — (Препринт ; arXiv:1705.02092v1). — Режим доступу: https://arxiv.org/abs/1705.02092 (дата звернення: 30.04.2021). — Назва з екрана.

18. Neural Style Transfer: A Review [Електронний ресурс] / Yongcheng Jing [та ін.]. — [Б. м. : б. в.], 2018. — 25 с. — (Препринт ; arXiv:1705.04058v7). — Режим доступу: https://arxiv.org/abs/1705.04058v7 (дата звернення: 30.04.2021). — Назва з екрана.

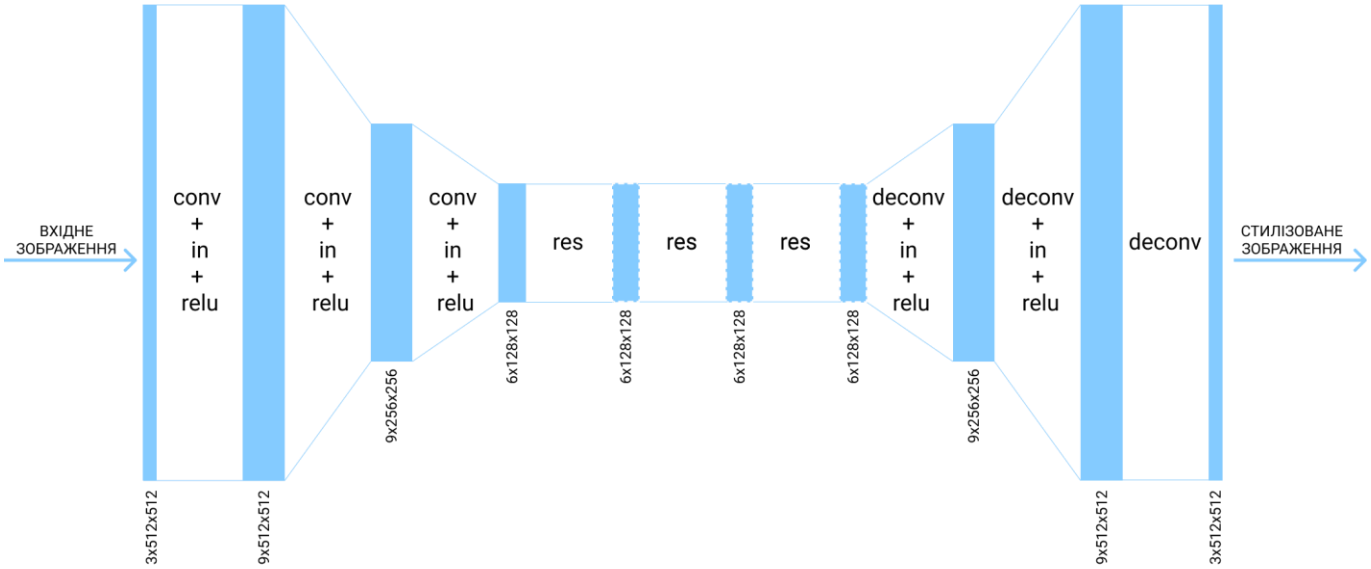
19. Ronneberger O. U-Net: Convolutional Networks for Biomedical Image Segmentation [Електронний ресурс] / Olaf Ronneberger, Philipp Fischer, Thomas Brox. — [Б. м. : б. в.], 2015. — 8 с. — (Препринт ; arXiv:1505.04597v1). — Режим доступу: https://arxiv.org/abs/1505.04597 (дата звернення: 30.04.2021). — Назва з екрана.

20. Ulyanov D. Instance Normalization: The Missing Ingredient for Fast Stylization [Електронний ресурс] / Dmitry Ulyanov, Andrea Vedaldi, Victor Lempitsky. — [Б. м. : б. в.], 2017. — 6 с. — (Препринт ; arXiv:1607.08022v3). — Режим доступу: https://arxiv.org/pdf/1607.08022.pdf (дата звернення: 30.04.2021). — Назва з екрана.

Додаток А

(обов’язковий)

Архітектура мережі передачі стилю



Додаток Б

(обов’язковий)

Результати перенесення стилю на мережах створених у Create ML

Додаток В

(обов’язковий)

Результати перенесення стилю на мережах створених у PyTorch

Додаток Г

(обов’язковий)

Інтерфейс створеного застосунку